SISTEMI OPERATIVI

ESERCIZIO N. 1 del 17 LUGLIO 2000

Ad un matrimonio, gli **invitati** portano doni agli sposi. Gli invitati sono di tre tipi: i **parenti**, gli **amici dello sposo** e le **amiche della sposa**. I **parenti** degli sposi vogliono portare il loro dono ad entrambi gli sposi insieme; gli **amici dello sposo** vogliono portare il loro dono solo allo sposo e le **amiche della sposa** vogliono portare il loro dono solo alla sposa. I parenti hanno la precedenza sugli amici e sulle amiche.
Si implementi una soluzione usando il costrutto monitor per modellare la coppia di sposi e i processi per modellare i tre tipi di invitati, e si descriva la sincronizzazione tra i processi. Nella soluzione si massimizzi l'utilizzo delle risorse. Si discuta se la soluzione proposta può presentare starvation e in caso positivo per quali processi, e si propongano modifiche e/o aggiunte per evitare starvation.

Versione 1 con **while**

```
program matrimonio

type tipo = (parente, amico_sposo, amica_sposa);

type invitato = process(t: tipo)
begin
      coppia.saluta(t);
      <saluta gli sposi>
      coppia.fine_saluto(t);
end

type coppia_di_sposi = monitor

{ variabili del monitor }
var  sposo_libero, sposa_libera: boolean;
      coda_parenti, coda_lui, coda_lei: condition;

procedure entry saluta(t: tipo)
begin
  if t = parente then
  begin
    while (sposo_libero <> true) or (sposa_libera <> true) do
      coda_parenti.wait;
    sposo_libero := false; sposa_libera := false;
  end;

  else if t = amico_sposo then
  begin
    while sposo_libero <> true or coda_parenti.queue do
      coda_lui.wait;
    sposo_libero := false;
  end;
```

```
  else  { t = amica_sposa }
  begin
    while sposa_libera <> true or coda_parenti.queue do
      coda_lei.wait;
    sposa_libera := false;
  end;

end


procedure fine_saluto(t: tipo)
begin
  if t = parente then
    sposo_libero := true; sposa_libera := true;
  else if t = amico_sposo then
    sposo_libero := true;
   else  { t = amica_sposa }
    sposa_libera := true;

  coda_parenti.signal;
  coda_lui.signal;
  coda_lei.signal;
end

begin { inizializzazione dellevariabili }
       sposo_libero := true;
       sposa_libera := true;
end

var  coppia: coppiadisposi; { il nostro monitor }
       parente1, parente2, ... : invitato(parente);
       amico1, amico2, ... : invitato(amico_sposo);
       amica1, amica2, ... : invitato(amica_sposa);

begin end.
```

Versione 2 con **if**

```
program matrimonio

type tipo = (parente, amico_sposo, amica_sposa);

type invitato = process(t: tipo)
begin
      coppia.saluta(t);
      <saluta gli sposi>
      coppia.fine_saluto(t);
end

type coppia_di_sposi = monitor

{ variabili del monitor }
var  sposo_libero, sposa_libera: boolean;
      coda_parenti, coda_lui, coda_lei: condition;

procedure saluta(t: tipo)
begin
  if t = parente then
  begin
    if (sposo_libero <> true) or (sposa_libera <> true) then
      coda_parenti.wait;
    sposo_libero := false; sposa_libera := false;
  end;

  else if t = amico_sposo then
  begin
    if sposo_libero <> true or coda_parenti.queue then
      coda_lui.wait;
    sposo_libero := false;
  end;
```

```
  else  { t = amica_sposa }
  begin
    if sposa_libera <> true or coda_parenti.queue then
      coda_lei.wait;
    sposa_libera := false;
  end;

end


procedure fine_saluto(t: tipo)
begin
  if t = parente then
    sposo_libero := true; sposa_libera := true;
  else if t = amico_sposo then
    sposo_libero := true;
   else  { t = amica_sposa }
    sposa_libera := true;

  if (sposo_libero = true) and (sposa_libera = true) and
coda_parenti.queue then
    coda_parenti.signal;
  else
  begin
    if sposo_libero = true then
      coda_lui.signal;
    if sposa_libera = true then
      coda_lei.signal;
  end;
end

begin { inizializzazione delle variabili }
       sposo_libero := true;
       sposa_libera := true;
end
```

```
var  coppia: coppiadisposi; { il nostro monitor }
     parente1, parente2, ... : invitato(parente);
     amico1, amico2, ... : invitato(amico_sposo);
     amica1, amica2, ... : invitato(amica_sposa);

begin end.
```

Considerazioni

Nella soluzione con l'**if** si nota che l'invitato che risveglia deve testare le condizioni, mentre nella soluzione con il **while**, l'invitato risveglia incondizionatamente un invitato per coda, il quale sarà in carico di ritestare le condizioni.

Starvation

Le soluzioni proposte presentano starvation, infatti è possibile che un tipo di processi monopolizzi la coppia di sposi. In particolare la coppia è contesa tra i parenti e gli amici.
La starvation è in parte attenuata dal fatto che vengano risvegliati per primi i parenti, che sono i processi che necessitano di maggiori risorse.

Un modo per ridurre ulteriormente la starvation è quello di utilizzare dei contatori, in modo che dopo S saluti di parenti/amici, la coppia venga lasciata agli amici/parenti.

Infine si potrebbe associare un numero progressivo ad ogni invitato che arriva, e rispettare rigorosamente l'ordine d'arrivo per i saluti. Questo però sarebbe penalizzante in termini di utilizzo delle risorse; ad esempio, non permetterebbe ad una amica di salutare la sposa contemporaneamente ad un amico che saluta lo sposo, se c'è un invitato di tipo parenti prima di lei.